Team :- 4 Review 1

Admin

Bipin Nayak : 24SCSE1180206 bipin.24scse1180206@galgotiasuniversity.ac.in
Github ID : https://github.com/BipinNayak0146

Team Members:-

Deepak Singh : 24SCSE1180234 deepak.24scse1180234@galgotiasuniversity.ac.in
Github ID : https://github.com/Deepak077x
MD Dilkhush : 24SCSE1180481 md.24scse1180481@galgotiasuniversity.ac.in
Github ID :
Neeraj Rai : 24SCSE1180566 neeraj.24scse1180466@galgotiasuniversity.ac.in
Github ID : https://github.com/Neerajrai-wq

Project Title

Hostel Mess Feedback & Voting System

A console-based Java application for student feedback and meal preference voting.

1. Project Description :

The Mess Feedback System is a Java-based console application designed to allow students to provide feedback for hostel mess meals and for admins to review all submitted feedback. The application uses basic File I/O operations to store and retrieve data from text files (users.txt and feedbacks.txt) and demonstrates principles of object-oriented programming, file handling, and menu-driven logic.

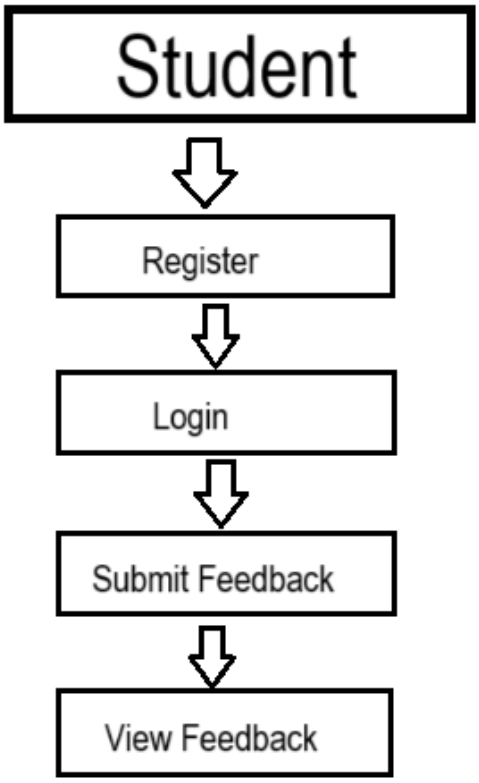
2. Objective

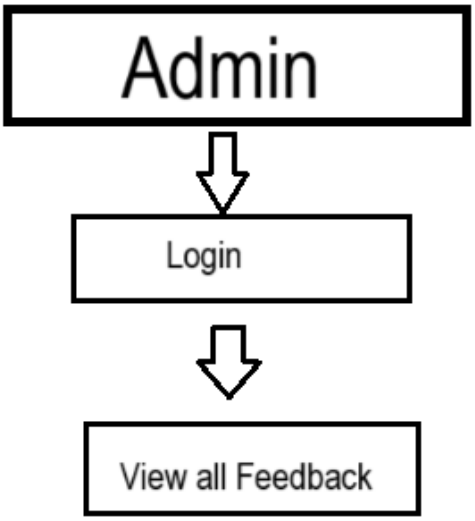
To build a simple and effective feedback management system for hostel mess using core Java without
databases. The system will allow:
- User registration and login
- Role-based menu access (student/admin)
- Feedback submission with date and rating
- Viewing of submitted feedback (all or by user)

3. Technologies and Tools Used

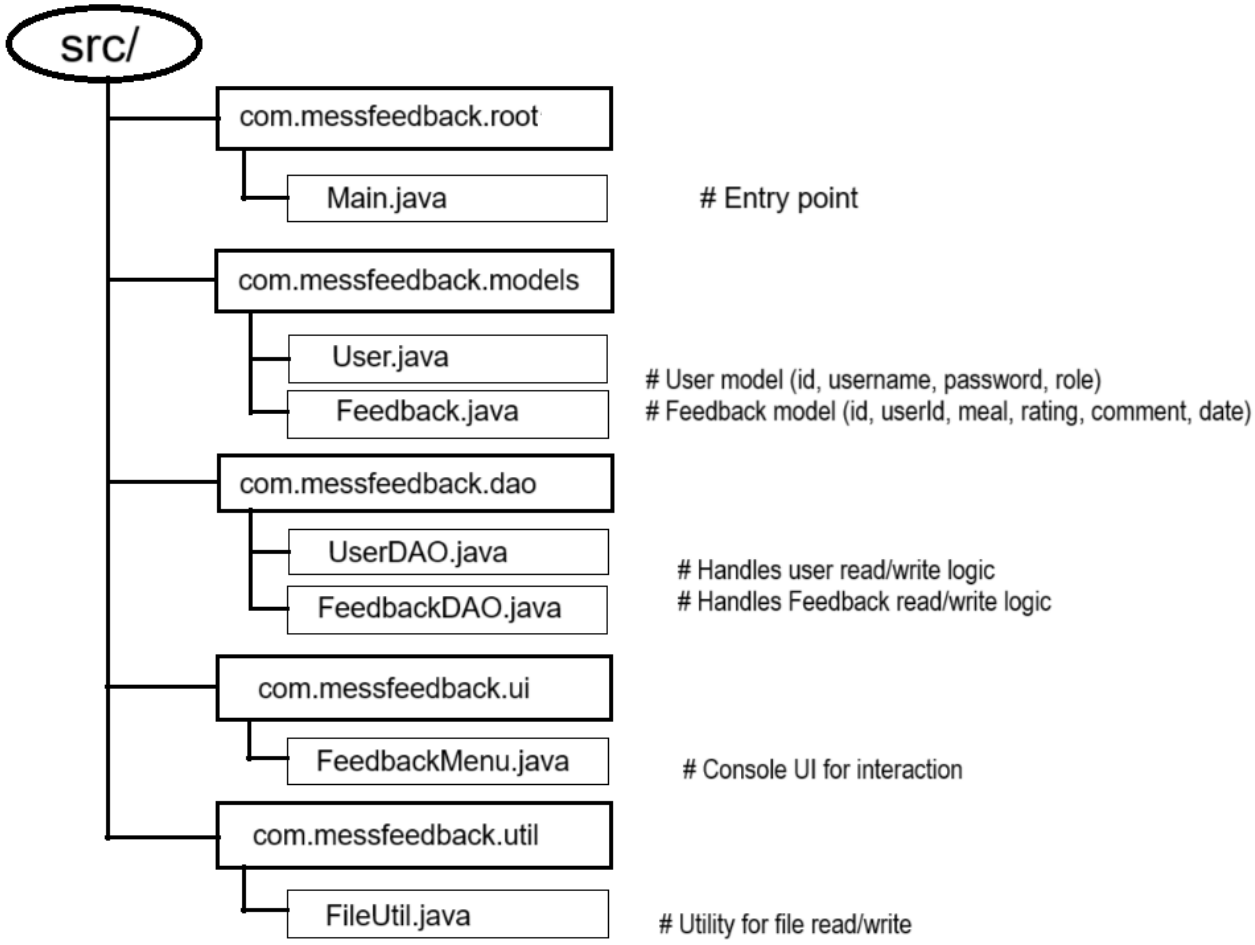
- Language: Java (JDK 17 or above)
- IDE/Text Editor: IntelliJ IDEA / Notepad
- File Handling: Text files (.txt) using Java I/O
- Architecture: Layered (Model, DAO, UI, Util)

4. Functional Requirements





5. Project Structure



➢ **models package –**

- **Class: Feedback.java**
  - This is your **data model class**, containing:
    - int id
    - String studentName
    - String mealType
    - int rating
    - String comments
    - String date

➢ **dao package –**

- **Class: FeedbackDAO.java**
  - Handles **file operations (CRUD)**:

- addFeedback(Feedback fb)
- List<Feedback> getAllFeedbacks()
- boolean updateFeedback(int id, Feedback updatedFb)
- boolean deleteFeedback(int id)
- Uses FileWriter, BufferedReader, etc.

➢ **<u>util package –</u>**

🔹 **Class: FileUtil.java**

- Reusable utility functions:
  - generateId() (returns new feedback ID)
  - String toCSV(Feedback fb)
  - Feedback fromCSV(String line)

➢ **ui package –**

🔹 **Class: FeedbackMenu.java**

- Contains your **Scanner-based menu**:
  - showMainMenu()
  - Handles input, calls DAO methods, validates user input.

➢ **Root Package –**

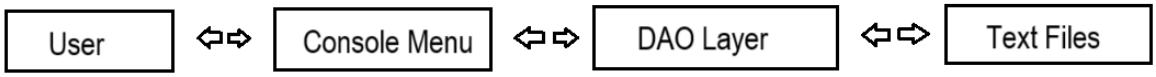
🔹 **Class: Main.java**

- Just contains the main() method:
  - Calls new FeedbackMenu().showMainMenu();

📦 **Summary of Class Structure:**

Package	Class Name	Purpose
models	Feedback	Holds feedback fields (data model)
dao	FeedbackDAO	Handles reading/writing to the file
util	FileUtil	Optional: ID generation & CSV helpers
ui	FeedbackMenu	Menu UI with Scanner
(root)	Main	Starts the application

6. Data Flow Diagram



7. Sample File Structure

users.txt:
1,Bipin Nayak, Bipin@123, student
2,Deepak Singh,Deepak@123,admin

feedbacks.txt:
Feedback{feedbackId=1, userId=0, mealType='Breakfast', rating=1, comments='Its good in taste', date='2025-05-24'}

8. Screenshots

Registration

```
==== Mess Feedback System ====
1. Register
2. Login
3. Exit
Select an option: 1
Enter username: Bipin Nayak
Enter password: Bipin@123
Enter role (student/admin): student
Registration failed. Username might already exist.
```

Login

```
==== Mess Feedback System ====
1. Register
2. Login
3. Exit
Select an option: 2
Enter username: Bipin Nayak
Enter password: Bipin@123
Login successful!
```

Feedback submission

```
Welcome, Bipin Nayak
1. Submit Feedback
2. View My Feedback
3. Logout
Choice: 1
Meal Type (Breakfast/Lunch/Dinner): Lunch
Rating (1-5): 4
Comments: Its little bit good not bad in taste
Feedback submitted successfully.
```

Student view there submission

```
Welcome, Bipin Nayak
1. Submit Feedback
2. View My Feedback
3. Logout
Choice: 2
Your Feedbacks:
Feedback{feedbackId=1, userId=0, mealType='Breakfast', rating=1, comments='Its good in taste', date='2025-05-24'}
Feedback{feedbackId=2, userId=0, mealType='Lunch', rating=4, comments='Its little bit good not bad in taste', date='2025-05-24'}
```

Student logout

```
Welcome, Bipin Nayak
1. Submit Feedback
2. View My Feedback
3. Logout
Choice: 3
```

Admin login

```
==== Mess Feedback System ====
1. Register
2. Login
3. Exit
Select an option: 2
Enter username: Deepak Singh
Enter password: Deepak@123
Login successful!
```

Admin View

```
Welcome Admin, Deepak Singh
1. View All Feedback
2. Logout
Choice: 1
All Feedbacks:
Feedback{feedbackId=1, userId=0, mealType='Breakfast', rating=1, comments='Its good in taste', date='2025-05-24'}
Feedback{feedbackId=2, userId=0, mealType='Lunch', rating=4, comments='Its little bit good not bad in taste', date='2025-05-24'}
```

Admin Logout

```
Welcome Admin, Deepak Singh
1. View All Feedback
2. Logout
Choice: 2
```

9.Known Issue: Input Mismatch Exception

During the program execution, if the user **enters an invalid input type** (e.g., typing a word like "register" instead of the number 1), the application throws the following exception:

```
==== Mess Feedback System ====
1. Register
2. Login
3. Exit
Select an option: register
Exception in thread "main" java.util.InputMismatchException Create breakpoint
    at java.base/java.util.Scanner.throwFor(Scanner.java:964)
    at java.base/java.util.Scanner.next(Scanner.java:1619)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2284)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2238)
    at com.messfeedback.ui.FeedbackMenu.start(FeedbackMenu.java:31)
    at com.messfeedback.Main.main(Main.java:11)
```

- **Reason:**

The application expects numeric input using Scanner.nextInt() for menu selections (like 1, 2, 3). If the user types text such as "register" instead of selecting a number, it causes a **type mismatch**.

- **How It Can Be Handled (Future Scope):**

- Add **input validation** using Scanner.hasNextInt() before reading an integer.
- Wrap nextInt() in a try-catch block to handle incorrect input and prompt the user again instead of crashing.

- **Temporary User Instruction:**

For stable operation, users should **enter numeric options (1, 2, 3, etc.)** exactly as shown in the menu.

10.Challenges Faced

- Handling input mismatch exceptions using Scanner.nextInt()

- Ensuring proper file format and parsing comments containing commas

- Assigning unique feedback IDs based on file contents

11. Conclusion

This project helped in learning:

- Java File I/O operations (read/write text files)

- Modular programming using models and DAOs

- Exception handling and user input validation

- Implementing real-world logic using simple console interaction